AMENDMENTS IN THE CLAIMS

1. (currently amended) A method for full speculation of instruction processing in a multiprocessor data processing system comprising:

issuing from a processor a barrier operation on a system bus of said data processing system; and

speculatively executing operations associated with instructions sequentially following said barrier operation in an instruction sequence prior to completion of said barrier operation, wherein operations are speculatively executed relative to a preceding barrier operation whenever the preceding barrier operation has not yet completed.

2. (currently amended) The method of Claim 1, ~~wherein said executing step executes said operations, prior to said issuing step~~ further comprising immediately forwarding data returned by a load instruction among said instructions sequentially following said barrier operation to execution units of said processor, wherein said data is utilized within said operations, where necessary, regardless of a completion status of said barrier operation, wherein further, said operations and results/data generated by said operations are tagged as barrier-speculative when said operations are speculatively executed relative to the barrier operation.

3. (currently amended) The method of Claim ~~1~~ 2, wherein said executing step further comprises:

issuing a load request for data;

responsive to a return of said data, immediately forwarding said data to a register of said processor; and

providing said data to subsequent processes that utilize said data.

4. (currently amended) The method of claim 3, wherein tagging of processes and results/data includes: ~~further comprising~~ setting a flag within said register when said barrier operation has not yet completed, wherein said flag indicates that said load instruction, returned data and each instruction executed and each result generated by said subsequent processes and stored within said register is barrier-speculative, pending a completion of said barrier operation.

5. (currently amended) The method of Claim 4, further comprising:

monitoring for said completion of said barrier operation without an invalidate response or incorrect dependency;

responsive to said completion, resetting said flag, wherein ~~and concurrently indicating~~ said register is no longer tagged as barrier-speculative and said processes are allowed to continue executing as non-speculative.

when the data exhibits incorrect dependency or an invalidate response is received, discarding the data and other values from the registers dependent on the data and re-executing each corresponding operation.

6. (currently amended) The method of Claim 5, wherein further, when an invalidate is received prior to said completion, said processor discards said data and each of said results from said register.

7. (currently amended) The method of Claim 6, wherein said operations include load requests and branch instructions, and wherein further said method provides embedded branch speculation within said operations and speculative load request, relative to a preceding barrier operation issued ~~issuing~~ within a branch path.

8. (currently amended) A multiprocessor computer system comprising:

a plurality of processors interconnected by a system bus, wherein said processors including a first processor that speculatively issues load requests and speculatively processes subsequent instructions utilizing data returned by said load request before a completion of a barrier operation that is issued sequentially before said load requests and subsequent instructions ~~in an instruction sequence~~, wherein said load request is speculatively issued and said subsequent instructions are speculatively processed relative to the barrier operation such that the speculative issuance and speculative processing are barrier-speculative; and

a memory hierarchy connected to said plurality of processors via said system bus that sources said data.

9. (original) The multiprocessor computer system of Claim 8, wherein said first processor comprises a load/store unit with logic that controls issuing of load and store instructions before completion of a preceding barrier operation to provide said data to a register of said first processor prior to a return of an acknowledgment for said preceding barrier operations.

10. (currently amended) The multiprocessor computer system of claim 8, wherein said first processor further comprises:

execution units that processes instructions that utilize said data when said data is placed in said register; and

logic, affiliated with said register, that sets a flag within said register when a value resulting from executing said instructions is placed in said register prior to said completion, wherein said flag messages to the execution units that said instruction and said results are barrier-speculative, pending a completion of said barrier operation.

11. (currently amended) The multiprocessor computer system of claim ~~8~~ 10, wherein said logic further:

resets said flag responsive to said completion; and

when the data exhibits incorrect dependency or an invalidate response is received, discarding the data and other values from the registers dependent on the data and re-executing each corresponding operation.

12. (original) The multiprocessor computer system of claim 11, wherein said first processor further comprises a plurality of execution queues and logic for setting a bit associated with an entry of said queues to indicate whether an instruction placed in said entry is speculative with respect to said barrier operation.

13. (original) The multiprocessor computer system of claim 11, wherein said first processor further comprises a plurality of execution queues and logic for setting a bit associated with an entry of said queues to indicate whether an instruction placed in said entry is speculative

with respect to an unresolved branch instruction that precedes said instruction in said instruction sequence

14. (currently amended) A processor comprising:

a plurality of execution units including a load/store unit, wherein said load/store unit speculatively executes load requests and ~~offer other execution into speculative execute~~ other instructions before completion of a barrier operation that precedes said load requests and other instructions in an instruction sequence;

a rename register that includes a plurality of entries, wherein each entry has a barrier-speculation flag and an associated general purpose register identifier; and

logic for setting said barrier-speculation flag to indicate when a value stored in said entry is speculative, ~~pending~~ relative to completion of said barrier operation.

15. (original) The processor of Claim 14, wherein said load/store unit provides data returned by said load requests immediately to an entry of said rename register for utilization within subsequent processes that require said data.

16. Canceled

17. (currently amended) The processor of Claim ~~16~~ 14, wherein~~,~~;

said load/store unit messages said execution units and said logic when said barrier operation completes; and

said logic, responsive to a receipt of a message indicating successful completion of said barrier operation, resets each flag associated with a register entry that was speculative with respect to said barrier operation.

18. (currently amended) The processor of Claim ~~17~~ 14, further comprising:

a plurality of issue queues associated with said execution units in which instructions to be executed are placed; and

logic for indicating that a particular instruction within one of said issue queues is speculative with respect to the barrier operation.

19. (currently amended) The processor of Claim ~~17~~ 14, further comprising:

a plurality of issue queues associated with said execution units in which instructions to be executed are placed; and

logic for indicating that a particular instruction within one of said issue queues is speculative with respect to an unresolved branch instruction that precedes said instruction within said instruction sequence.

20. (original) The processor of Claim 18, further comprising:

an enhanced internal instruction set architecture that includes a setable bit, which indicates whether an instruction is speculative, wherein said logic sets said setable bit responsive to whether said barrier operation has completed; and

when said barrier operation has completed, said logic resets said bit.

21. (currently amended) The processor of Claim 18, wherein said issue queues includes a speculation bit associated with each entry location, wherein said speculation bit is set by said logic when said particular instruction is placed in an associated entry location, and reset only when said barrier operation has successfully completed.

22. (currently amended) A data processing system comprising:

a memory; and

at least two processors interconnected to each other and said memory via a system bus, wherein a first processor comprises:

a plurality of execution units including a load/store unit, wherein said load/store unit speculatively executes load requests and ~~offer other execution into speculative execute~~ other instructions before completion of a barrier operation that precedes said load requests and other instructions in an instruction sequence;

a rename register that includes a plurality of entries, wherein each entry has a barrier-speculation flag and an associated general purpose register identifier; and

logic for setting said barrier-speculation flag to indicate when a value stored in said entry is speculative, ~~pending~~ relative to completion of said barrier operation.

23. (currently amended) The data processing system of Claim 22, wherein said load/store unit provides data returned by said load requests immediately to an execution unit of said processor for utilization within subsequent processes that require said data, wherein results generated by said subsequent processes are also tagged as barrier-speculative when said processes complete prior to completion of said barrier operation.

24. Canceled

25. (currently amended) The data processing system of Claim ~~24~~ 23, wherein~~,~~:

said load/store unit messages said execution units and said logic when said barrier operation completes; and.

said logic, responsive to a receipt of a message indicating successful completion of said barrier operation, resets each flag associated with a register entry that was speculative with respect to said barrier operation.

26. (original) The data processing system of Claim 25, further comprising:

a plurality of issue queues associated with said execution units in which instructions to be executed are placed; and

logic for indicating that a particular instruction within one of said issue queues is speculative with respect to the barrier operation.

27. (original) The data processing system of Claim 26, further comprising:

an enhanced internal instruction set architecture that includes a setable bit, which indicates whether an instruction is speculative, wherein said logic sets said setable bit responsive to whether said barrier operation has completed; and

when said barrier operation has completed, said logic resets said bit.

28. (original) The data processing system of Claim 26, wherein said issue queues includes a speculation bit associated with each entry location, wherein said speculation bit is set by said logic when said particular instruction is placed in an associated entry location, and reset only when said barrier operation has successfully completed.

29. (New) The method of Claim 7, further comprising:

setting a first bit within said register to indicate said load operation is being issued within a speculative branch; and

setting a second bit within said register to indicate that said load operation is being issued speculatively, relative to a preceding barrier operation within the speculative branch.

30. (New) The multiprocessor computer system of Claim 8, wherein said first processor further comprises:

a branch processing unit that executes branch instructions within the sequence of instructions;

execution logic that speculatively selects a branch prior to resolution of a branch instruction, wherein, when said barrier operation is sequentially ahead of the load requests in execution sequence, said execution logic further includes:

means for providing barrier-speculation of said load request within the speculative branch path, wherein said load request is issued speculatively, relative to the preceding barrier operation, and within the speculative branch path;

wherein further said execution logic includes logic for:

setting a first bit within said register to indicate said load operation is being issued within a speculative branch; and

setting a second bit within said register to indicate that said load operation is being issued speculatively, relative to a preceding barrier operation and within the speculative branch.